AMENDMENTS TO THE CLAIMS

Please cancel claims 1-2, 16, 18-19, 33-35 and 49 and amend the remaining claims as follows.

1. (Canceled)

2. (Canceled)

3. (Currently amended) [The method of claim 1] A method of joining a first table t_1 and a second table t_2, each table containing rows and columns and being divided into one or more partitions, the method including:

(a) calculating a correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2;

(b) receiving a query requesting a join between table t_1 and table t_2; and

(c) performing a joining algorithm, wherein the partitions containing the rows to be joined are determined based at least in part upon the correlation function,

wherein the joining algorithm comprises:

(c1) calculating, based at least in part upon the correlation function, a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined;

(c2) determining, based at least in part upon the correlation function, a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2;

(c3) joining a set of f_2 partitions of table t_2 starting at P_2 with a set of f_1 partitions of table t_1 starting at P_1;

(c4) updating P_1 and P_2.

4. (Original) The method of claim 3, wherein (c3) and (c4) are repeated while at least one table has at least one active, non-eliminated partition.

5. (Original) The method of claim 3, wherein the f_2 partitions to be joined in (c3) are contiguous.

6. (Original) The method of claim 3, wherein the f_1 partitions to be joined in (c3) are contiguous.

7. (Original) The method of claim 3, wherein the f_2 partitions to be joined in (c3) are not contiguous.

8. (Original) The method of claim 3, wherein the f_1 partitions to be joined in (c3) are not contiguous.

9. (Original) The method of claim 3, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in (c3) are increased, the method further comprises:

(c31) setting a parameter eps equal to the minimum number of inactive or eliminated partitions in (i) the span of f_1 partitions in table t_1 beginning at P_1 and in (ii) the span of f_2 partitions of table t_2 beginning at P_2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

10. (Original) The method of claim 9, wherein (c31), (c32), and (c33) are repeated if some of the partitions added in the preceding iteration of (c31), (c32), and (c33) are empty.

11. (Original) The method of claim 3, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in (c3) is increased, the method further comprises:

(c31) setting a parameter eps equal to the result of the function FLOOR(x/2), wherein x is a sum of the number of inactive or eliminated partitions in the span of f_1 partitions in table t_1 beginning at P_1 and the span of f_2 partitions in table t_2 beginning at P_2, and FLOOR(x/2) returns a largest integer that is less than or equal to x/2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

12. (Original) The method of claim 11, wherein (c31), (c32), and (c33) are repeated if some of the partitions added in the preceding iteration of (c31), (c32), and (c33) are empty.

13. (Currently amended) [The method of claim 1] A method of joining a first table t_1 and a second table t_2, each table containing rows and columns and being divided into one or more partitions, the method including:

(a) calculating a correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2;

(b) receiving a query requesting a join between table t_1 and table t_2; and

(c) performing a joining algorithm, wherein the partitions containing the rows to be joined are determined based at least in part upon the correlation function,

wherein calculating the correlation function includes:

joining table t_1 to table t_2 using PK=FK as the join condition to produce a join result having rows, each row including a value from cv_1 and a value from cv_2, wherein PK denotes a primary key column in table t_1, FK denotes a foreign key column in table t_2, cv_1 denotes a first correlated value column in table t_1, and cv_2, denotes a second correlated value column in table t_2;

creating an initial running constraint (RC), the initial running constraint comprising a null range; and

producing a derived constraint rule (DCR) having the following form:

$(PK = FK) \rightarrow cv_2 + c_1 \leq cv_1 \leq cv_2 + c_2$,

where c_1 and c_2 are constants, and "$\rightarrow$" means "implies;"

by performing the following processing for each row in the join result:

computing a new constraint (NEW), having a range; and

modifying RC by merging the range of NEW with the range of RC.

14. (Original) The method of claim 13, wherein the joining algorithm comprises:

(c1) calculating, based at least in part upon the correlation function, a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined, wherein:

p is set equal to a size of a partition range of table t_1 and table t_2;

pc_1 is set equal to the value of (SIGN(c_1)*CEILING(ABS(c_1)/p)), wherein SIGN(c_1) returns a value of –1 if c_1 is less than zero, otherwise SIGN(c_1) returns a value of 1, ABS(c_1) returns an absolute value of c_1, and CEILING(ABS(c_1)/p) returns a smallest integer that is not less than the value of ABS(c_1)/p;

pc_2 is set equal to the value of (SIGN(c_2)*CEILING(ABS(c_2)/p)), wherein SIGN(c_2) returns a value of –1 if c_2 is less than zero, otherwise SIGN(c_2) returns a value of 1, ABS(c_2) returns an absolute value of c_2, and CEILING(ABS(c_2)/p) returns a smallest integer that is not less than the value of ABS(c_2)/p;

n is set equal to $pc_2 - pc_1 + 1$, wherein n is a number of contiguous partitions in table t_1 that may have rows matching rows in a single partition of table t_2;

m is a maximum number of file contexts;

f_2 is set equal to the smallest integer value that is equal to or greater than the value of ((m-n)/2);

f_1 is set equal to $n + f_2 - 1$;

(c2) determining, based at least in part upon the correlation function, a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2, wherein:

P_2 is set equal to a lowest partition number in table t_2 such that P_2 is a first active, non-eliminated partition in table t_2, and at least one of the partitions in the interval between $P_2 - pc_2$ and $P_2 - pc_1$ in table t_1 is an active, non-eliminated partition;

P_1 is set equal to $P_2 - pc_2$;

(c3) joining a set of f_2 partitions of table t_2 starting at P_2 with a set of f_1 partitions of table t_1 starting at P_1;

(c4) updating P_1 and P_2, wherein:

P_2 is set equal to a lowest partition number P_2^* in table t_2, wherein:

the lowest partition number P_2^* is greater than or equal to the sum of P_2 + f_2;

P_2^* is a first active, non-eliminated partition;

at least one of the partitions in the interval between $P_2^* - pc_2$ and $P_2^* - pc_1$ in table t_1 is an active, non-eliminated partition; and

P_1 is set equal to $P_2^* - pc_2$.

15. (Original) The method of claim 14, wherein (c3) and (c4) are repeated while at least one table has at least one active, non-eliminated partition.

16. (Canceled)

17. (Original) A method of joining a first table t_1 and a second table t_2, each table containing rows and columns and being divided into one or more partitions, the method including:

(a) calculating a correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2, wherein calculating the correlation function includes:

joining table t_1 to table t_2 using PK=FK as the join condition to produce a join result having rows, each row including a value from cv_1 and a value from cv_2, wherein PK denotes a primary key column in table t_1, FK denotes a foreign key column in table t_2, cv_1 denotes a first correlated value column in table t_1, and cv_2, denotes a second correlated value column in table t_2;

creating an initial running constraint (RC), the initial running constraint comprising a null range;

producing a derived constraint rule (DCR) having the following form:

$(PK = FK) \rightarrow cv_2 + c_1 \leq cv_1 \leq cv_2 + c_2$,

where c_1 and c_2 are constants, and "→" means "implies;"

by performing the following processing for each row in the join result:

computing a new constraint (NEW), having a range; and

modifying RC by merging the range of NEW with the range of RC;

setting p equal to a size of a partition range of table t_1 and table t_2;

determining a first correlation coefficient pc_1 which is equal to the value of (SIGN(c_1) * CEILING(ABS(c_1)/p)), wherein SIGN(c_1) returns a value of -1 if c_1 is less than zero, otherwise SIGN(c_1) returns a value of 1, ABS(c_1) returns the absolute value of c_1, and CEILING(ABS(c_1)/p) returns a smallest integer that is equal to or greater than ABS(c_1)/p;

determining a second correlation coefficient pc_2 which is equal to the value of (SIGN(c_2) * CEILING(ABS(c_2)/p)), wherein SIGN(c_2) returns a value of -1 if c_2 is less than zero, otherwise SIGN(c_2)

returns a value of 1, ABS(c_2) returns the absolute value of c_2, and CEILING(ABS(c_2)/p) returns a smallest integer that is equal to or greater than ABS(c_2)/p;

(b) receiving a query requesting a join between table t_1 and table t_2;

(c) calculating a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined, wherein calculating f_1 and f_2 include:

setting n equal to $pc_2 - pc_1 + 1$;

determining a parameter m, which is a maximum number of file contexts;

setting f_2 equal to the smallest integer value that is equal to or greater than the value of ((m-n)/2); and

setting f_1 equal to $n + f_2 - 1$;

(d) determining a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2, wherein P_1 and P_2 are calculated by;

setting P_2 equal to a lowest partition number in t_2 such that P_2 is a first active, non-eliminated partition in table t_2, and at least one of the partitions in the interval between $P_2 - pc_2$ and $P_2 - pc_1$ in table t_1 is an active, non-eliminated partition; and

setting P_1 equal to $P_2 - pc_2$;

(e) performing a joining algorithm, wherein a set of f_2 partitions of table t_2 starting at P_2 are joined with a set of f_1 partitions of table t_1 starting at P_1, wherein the joining algorithm includes:

creating a file context, which stores at least location data for a row and a first value associated with the row, for each partition of the set of partitions to be joined;

determining the lowest first value stored by the file contexts that is equal to or greater than a particular hash value; and

identifying rows with a particular first value by reading the file contexts;

(f) updating P_1 and P_2, wherein the updating P_1 and P_2 includes:

finding a lowest partition number P_2^* in t_2 that is greater than or equal to the sum of P2 + f2 such that P_2^* is a first active, non-eliminated

partition, and at least one of the partitions in the interval between $P_2^* - pc_2$ and $P_2^* - pc_1$ in table t_1 is an active, non-eliminated partition;

setting P_2 equal to P_2^*; and

setting P_1 equal to $P_2^* - pc_2$; and

(g) repeating steps (e)-(f) while at least one table has at least one active, non-eliminated partition.

18. (Canceled)

19. (Canceled)

20. (Currently amended) [The computer program of claim 18] A computer program, stored in tangible medium, for joining a first table t_1 and a second table t_2, each table containing rows and columns and being divided into one or more partitions, the program comprising executing instructions that cause a computer to:

(a) calculate a correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2;

(b) receive a query requesting a join between table t_1 and table t_2; and

(c) perform a joining algorithm, wherein the partitions containing the rows to be joined are determined based at least in part upon the correlation function,

wherein the joining algorithm comprises:

(c1) calculating, based at least in part upon the correlation function, a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined;

(c2) determining, based at least in part upon the correlation function, a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2;

(c3) joining a set of f_2 partitions of table t_2 starting at P_2 with a set of f_1 partitions of table t_1 starting at P_1;

(c4) updating P_1 and P_2.

21. (Original) The computer program of claim 20, wherein (c3) and (c4) are repeated while at least one table has at least one active, non-eliminated partition.

22. (Original) The computer program of claim 20, wherein the f_2 partitions to be joined in (c3) are contiguous.

23. (Original) The computer program of claim 20, wherein the f_1 partitions to be joined in (c3) are contiguous.

24. (Original) The computer program of claim 20, wherein the f_2 partitions to be joined in (c3) are not contiguous.

25. (Original) The computer program of claim 20, wherein the f_1 partitions to be joined in (c3) are not contiguous.

26. (Original) The computer program of claim 20, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in (c3) are increased, further comprising:

(c31) setting a parameter eps equal to the minimum number of inactive or eliminated partitions in (i) the span of f_1 partitions in table t_1 beginning at P_1 and in (ii) the span of f_2 partitions of table t_2 beginning at P_2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

27. (Original) The computer program of claim 26, wherein (c31), (c32), and (c33) are repeated if some of the partitions added in the preceding iteration of (c31), (c32), and (c33) are empty.

28. (Original) The computer program of claim 20, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in c3 is increased, the computer program further comprises:

(c31) setting a parameter eps equal to the result of the function FLOOR(x/2), wherein x is a sum of the number of inactive or eliminated partitions in the span of f_1 partitions in table t_1 beginning at P_1 and the span of f_2 partitions in table t_2 beginning at P_2, and FLOOR(x/2) returns a largest integer that is less than or equal to x/2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

29. (Original) The computer program of claim 28, wherein (c31), (c32), and (c33) are repeated if some of the partitions added in the preceding iteration of (c31), (c32), and (c33) are empty.

30. (Currently amended) [The computer program of claim 18] A computer program, stored in tangible medium, for joining a first table t_1 and a second table t_2, each table containing rows and columns and being divided into one or more partitions, the program comprising executing instructions that cause a computer to:

(a) calculate a correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2;

(b) receive a query requesting a join between table t_1 and table t_2; and

(c) perform a joining algorithm, wherein the partitions containing the rows to be joined are determined based at least in part upon the correlation function,

wherein calculating the correlation function includes:

joining table t_1 to table t_2 using PK=FK as the join condition to produce a join result having rows, each row including a value from cv_1 and a value from cv_2, wherein PK denotes a primary key column in table t_1, FK denotes a foreign key column in table t_2, cv_1 denotes a first correlated value column in table t_1, and cv_2, denotes a second correlated value column in table t_2;

creating an initial running constraint (RC), the initial running constraint comprising a null range; and

producing a derived constraint rule (DCR) having the following form:

$(PK = FK) \rightarrow cv_2 + c_1 \leq cv_1 \leq cv_2 + c_2$,

where c_1 and c_2 are constants, and "→" means "implies;"

by performing the following processing for each row in the join result:

computing a new constraint (NEW), having a range; and

modifying RC by merging the range of NEW with the range of RC.

31. (Original) The computer program of claim 30, wherein the joining algorithm comprises:

(c1) calculating, based at least in part upon the correlation function, a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined, wherein:

p is set equal to a size of a partition range of table t_1 and table t_2;

pc_1 is set equal to the value of (SIGN(c_1)*CEILING(ABS(c_1)/p)), wherein SIGN(c_1) returns a value of –1 if c_1 is less than zero, otherwise SIGN(c_1) returns a value of 1, ABS(c_1) returns an absolute value of c_1, and CEILING(ABS(c_1)/p) returns a smallest integer that is not less than the value of ABS(c_1)/p;

pc_2 is set equal to the value of (SIGN(c_2)*CEILING(ABS(c_2)/p)), wherein SIGN(c_2) returns a value of –1 if c_2 is less than zero, otherwise SIGN(c_2) returns a value of 1, ABS(c_2) returns an absolute value of c_2, and CEILING(ABS(c_2)/p) returns a smallest integer that is not less than the value of ABS(c_2)/p;

n is set equal to $pc_2 - pc_1 + 1$, wherein n is a number of contiguous partitions in table t_1 that may have rows matching rows in a single partition of table t_2;

m is a maximum number of file contexts;

f_2 is set equal to the smallest integer value that is equal to or greater than the value of ((m-n)/2);

f_1 is set equal to $n + f_2 - 1$;

(c2) determining, based at least in part upon the correlation function, a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2, wherein:

P_2 is set equal to a lowest partition number in table t_2 such that P_2 is a first active, non-eliminated partition in table t_2, and at least one of the partitions in the interval between $P_2 - pc_2$ and $P_2 - pc_1$ in table t_1 is an active, non-eliminated partition;

P_1 is set equal to $P_2 - pc_2$;

(c3) joining a set of f_2 partitions of table t_2 starting at P_2 with a set of f_1 partitions of table t_1 starting at P_1;

(c4) updating P_1 and P_2, wherein:

P_2 is set equal to a lowest partition number P_2^* in table t_2, wherein:

the lowest partition number P_2^* is greater than or equal to the sum of $P_2 + f_2$;

P_2^* is a first active, non-eliminated partition;

at least one of the partitions in the interval between $P_2^* - pc_2$ and $P_2^* - pc_1$ in table t_1 is an active, non-eliminated partition; and

P_1 is set equal to $P_2^* - pc_2$.

32. (Original) The computer program of claim 31, wherein (c3) and (c4) are repeated while at least one table has at least one active, non-eliminated partition.

33. (Canceled)

34. (Canceled)

35. (Canceled)

36. (Currently amended) [The system of claim 34] A system in which a first table t_1 is joined with a second table t_2, each table containing rows and columns and being divided into one or more partitions, comprising:

a massively parallel processing system comprising:

one or more nodes;

a plurality of CPUs, each of the one or more nodes providing access to one or more CPUs;

a plurality of processes each of the one or more CPUs providing access to one or more virtual processes;

each process configured to manage data, including the partitioned database table, stored in one of a plurality of data-storage facilities;

a partitioned table access component configured to select rows from the table by:

(a) calculating correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2;

(b) receiving a query requesting a join between table t_1 and table t_2; and

(c) performing a joining algorithm, wherein the partitions containing the rows to be joined are determined based at least in part upon the correlation function,

wherein the joining algorithm comprises:

(c1) calculating, based at least in part upon the correlation function, a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined;

(c2) determining, based at least in part upon the correlation function, a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2;

(c3) joining a set of f_2 partitions of table t_2 starting at P_2 with a set of f_1 partitions of table t_1 starting at P_1;

(c4) updating P_1 and P_2.

37. (Original) The system of claim 36, wherein (c3) and (c4) are repeated while at least one table has at least one active, non-eliminated partition.

38. (Original) The system of claim 36, wherein the f_2 partitions to be joined in (c3) are contiguous.

39. (Original) The system of claim 36, wherein the f_1 partitions to be joined in (c3) are contiguous.

40. (Original) The system of claim 36, wherein the f_2 partitions to be joined in (c3) are not contiguous.

41. (Original) The system of claim 36, wherein the f_1 partitions to be joined in (c3) are not contigu. The system of claim 36, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in (c3) are increased, further comprising:

(c31) setting a parameter eps equal to the minimum number of inactive or eliminated partitions in (i) the span of f_1 partitions in table t_1 beginning at P_1 and in (ii) the span of f_2 partitions of table t_2 beginning at P_2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

42. (Original) The system of claim 36, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in (c3) are increased, further comprising:

(c31) setting a parameter eps equal to the minimum number of inactive or eliminated partitions in (i) the span of f_1 partitions in table t_1 beginning at P_1 and in (ii) the span of f_2 partitions of table t_2 beginning at P_2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

43. (Original) The system of claim 42, wherein (c31), (c32), and (c33) are repeated if some of the partitions added in the preceding iteration of (c31), (c32), and (c33) are empty.

44. (Original) The system of claim 36, wherein the span of the f_1 partitions in table t_1 and the span of the f_2 partitions in table t_2 to be joined in c3 is increased, the system further comprises:

(c31) setting a parameter eps equal to the result of the function FLOOR(x/2), wherein x is a sum of the number of inactive or eliminated partitions in the span of f_1 partitions in table t_1 beginning at P_1 and the span of f_2 partitions in table t_2 beginning at P_2, and FLOOR(x/2) returns a largest integer that is less than or equal to x/2;

(c32) increasing the value of f_2 by eps;

(c33) increasing the value of f_1 by eps; and

(c34) after performing (c4), resetting the value of f_2 equal to the value of f_2 calculated in (c1) and resetting the value of f_1 equal to the value of f_1 calculated in (c1).

45. (Original) The system of claim 44, wherein (c31), (c32), and (c33) are repeated if some of the partitions added in the preceding iteration of (c31), (c32), and (c33) are empty.

46. (Currently amended) [The system of claim 34] A system in which a first table t_1 is joined with a second table t_2, each table containing rows and columns and being divided into one or more partitions, comprising:

a massively parallel processing system comprising:

one or more nodes;

a plurality of CPUs, each of the one or more nodes providing access to one or more CPUs;

a plurality of processes each of the one or more CPUs providing access to one or more virtual processes;

each process configured to manage data, including the partitioned database table, stored in one of a plurality of data-storage facilities;

a partitioned table access component configured to select rows from the table by:

(a) calculating correlation function between a first correlated value column of table t_1 and a second correlated value column of table t_2;

(b) receiving a query requesting a join between table t_1 and table t_2; and

(c) performing a joining algorithm, wherein the partitions containing the rows to be joined are determined based at least in part upon the correlation function,

wherein calculating the correlation function includes:

joining table t_1 to table t_2 using PK=FK as the join condition to produce a join result having rows, each row including a value from cv_1 and a value from cv_2, wherein PK denotes a primary key column in table t_1, FK denotes a foreign key column in table t_2, cv_1 denotes a first correlated value column in table t_1, and cv_2, denotes a second correlated value column in table t_2;

creating an initial running constraint (RC), the initial running constraint comprising a null range; and

producing a derived constraint rule (DCR) having the following form:

$(PK = FK) \rightarrow cv_2 + c_1 \leq cv_1 \leq cv_2 + c_2$,

where c_1 and c_2 are constants, and "→" means "implies;"

by performing the following processing for each row in the join result:

computing a new constraint (NEW), having a range; and

modifying RC by merging the range of NEW with the range of RC.

47. (Original) The system of claim 46, wherein the joining algorithm comprises:

(c1) calculating, based at least in part upon the correlation function, a first number f_1 and a second number f_2, wherein f_1 and f_2 denote the number of partitions of table t_1 and table t_2, respectively, to be joined, wherein:

p is set equal to a size of a partition range of table t_1 and table t_2;

pc_1 is set equal to the value of (SIGN(c_1)*CEILING(ABS(c_1)/p)), wherein SIGN(c_1) returns a value of –1 if c_1 is less than zero, otherwise SIGN(c_1) returns a value of 1, ABS(c_1) returns an absolute value of c_1, and CEILING(ABS(c_1)/p) returns a smallest integer that is not less than the value of ABS(c_1)/p;

pc_2 is set equal to the value of (SIGN(c_2)*CEILING(ABS(c_2)/p)), wherein SIGN(c_2) returns a value of –1 if c_2 is less than zero, otherwise SIGN(c_2) returns a value of 1, ABS(c_2) returns an absolute value of c_2, and CEILING(ABS(c_2)/p) returns a smallest integer that is not less than the value of ABS(c_2)/p;

n is set equal to $pc_2 - pc_1 + 1$, wherein n is a number of contiguous partitions in table t_1 that may have rows matching rows in a single partition of table t_2;

m is a maximum number of file contexts;

f_2 is set equal to the smallest integer value that is equal to or greater than the value of ((m-n)/2);

f_1 is set equal to $n + f_2 - 1$;

(c2) determining, based at least in part upon the correlation function, a first starting partition number P_1 for table t_1 and a second starting partition number P_2 for table t_2, wherein:

P_2 is set equal to a lowest partition number in table t_2 such that P_2 is a first active, non-eliminated partition in table t_2, and at least one of the partitions in the interval between $P_2 - pc_2$ and $P_2 - pc_1$ in table t_1 is an active, non-eliminated partition;

P_1 is set equal to $P_2 - pc_2$;

(c3) joining a set of f_2 partitions of table t_2 starting at P_2 with a set of f_1 partitions of table t_1 starting at P_1;

(c4) updating P_1 and P_2, wherein:

P_2 is set equal to a lowest partition number P_2^* in table t_2, wherein:

the lowest partition number P_2^* is greater than or equal to the sum of $P_2 + f_2$;

P_2^* is a first active, non-eliminated partition;

at least one of the partitions in the interval between $P_2^* - pc_2$ and $P_2^* - pc_1$ in table t_1 is an active, non-eliminated partition; and

P_1 is set equal to $P_2^* - pc_2$.

48. (Original) The system of claim 47, wherein (c3) and (c4) are repeated while at least one table has at least one active, non-eliminated partition.

49. (Canceled)